

Toast

Toast The <ava-toast> system provides a comprehensive notification solution with multiple variants, flexible positioning, smooth animations, and action buttons. Built around a service-based architecture, it offers both simple and advanced toast configurations for various user feedback scenarios.

How to use ■ `import { AavaToastService } from "@aava/play-core"; // Inject the service constructor (private toastService : AavaToastService) { } // Show different types of toasts`
`this . toastService . success ({ message : 'Operation completed!' }) ; this . toastService . error ({ message : 'Something went wrong!' }) ;`

Basic Usage ■ Simple toast notifications with default configurations.

Variants ■ Six distinct toast variants for different notification types and user feedback scenarios.

Available Variants ■ **Success** - Green styling for successful operations and confirmations **Error** - Red styling for errors and critical issues **Warning** - Orange/yellow styling for warnings and cautions **Info** - Blue styling for informational messages and tips **Default** - Neutral styling for general notifications **Custom** - Fully customizable styling and content

Sizes ■ Three size variants to accommodate different content lengths and interface requirements.

Available Sizes ■ **Small** - Compact size for minimal content and dense interfaces **Medium** - Standard size for most notification scenarios (default) **Large** - Prominent size for important messages and detailed content

Service Methods ■ The `ToastService` provides convenient methods for different toast types:

Success Toast ■ `this . toastService . success ({ title : "Success!" , message : "Operation completed successfully." , duration : 4000 , showCloseButton : true , }) ;`

Error Toast ■ `this . toastService . error ({ title : "Error!" , message : "Something went wrong. Please try again." , showRetryButton : true , retryButtonText : "Retry" , }) ;`

Warning Toast ■ `this . toastService . warning ({ title : "Warning" , message : "Please review the information carefully." , showActionButton : true , actionButtonText : "Review" , }) ;`

Info Toast ■ `this . toastService . info ({ title : "Information" , message : "Here is some important information." , showLearnMoreButton : true , learnMoreButtonText : "Learn More" , }) ;`

Custom Toast ■ `this . toastService . custom ({ title : "Custom Toast" , message : "This is a fully customizable toast." , customWidth : "500px" , customBackground : "#6366f1" , customTextColor : "#ffffff" , }) ;`

API Reference ■ **ToastService Methods** ■

Method	Parameters	Return Type	Description
<code>success()</code>	<code>SuccessToastConfig?</code>	<code>Promise<ToastResult></code>	Show success toast with green styling
<code>error()</code>	<code>ErrorToastConfig?</code>	<code>Promise<ToastResult></code>	Show error toast with red styling
<code>warning()</code>	<code>WarningToastConfig?</code>	<code>Promise<ToastResult></code>	Show warning toast with orange styling
<code>info()</code>	<code>InfoToastConfig?</code>	<code>Promise<ToastResult></code>	Show info toast with blue styling
<code>default()</code>	<code>DefaultToastConfig?</code>	<code>Promise<ToastResult></code>	Show default toast with neutral styling
<code>custom()</code>	<code>CustomToastConfig?</code>	<code>Promise<ToastResult></code>	Show custom toast with full customization

setPosition() `ToastPosition` void
Set global toast position

dismissAll() - void
Dismiss all active toasts

Toast Configuration Interfaces ■

Base ToastConfig	Property	Type	Default	Description
■	<code>title</code>	string?	"	Toast header text
	<code>message</code>	string?	"	Toast body text
	<code>duration</code>	number?	4000	Auto-dismiss duration in milliseconds
	<code>position</code>	<code>ToastPosition?</code>	'top-right'	Toast position on screen
■	<code>showCloseButton</code>	boolean?	true	Whether to

showCloseButton boolean? true Whether to show progress bar icon string? " Custom icon name iconColor string? " Custom icon color customWidth string? " Custom toast width customHeight string? " Custom toast height design 'classic' | 'modern' 'classic' Toast design variant size 'large' | 'medium' | 'small' 'large' Toast size variant SuccessToastConfig ■ Property Type Default Description type 'success' 'success' Toast type (always 'success') ErrorToastConfig ■ Property Type Default Description type 'error' 'error' Toast type (always 'error') showRetryButton boolean? false Whether to show retry button retryButtonText string? 'Retry' Text for retry button WarningToastConfig ■ Property Type Default Description type 'warning' 'warning' Toast type (always 'warning') showActionButton boolean? false Whether to show action button actionButtonText string? 'Action' Text for action button InfoToastConfig ■ Property Type Default Description type 'info' 'info' Toast type (always 'info') showLearnMoreButton boolean? false Whether to show learn more button learnMoreButtonText string? 'Learn More' Text for learn more button CustomToastConfig ■ Property Type Default Description type 'custom' 'custom' Toast type (always 'custom') customContent string? " Custom HTML content customBackground string? " Custom background color customTextColor string? " Custom text color progressColor string? " Custom progress bar color showCustomActions boolean? false Whether to show custom action buttons customActions CustomAction[]? " Array of custom action buttons Toast Position Options ■ Position Description 'top-left' Top-left corner of the screen 'top-center' Top-center of the screen 'top-right' Top-right corner of the screen (default) 'bottom-left' Bottom-left corner of the screen 'bottom-center' Bottom-center of the screen 'bottom-right' Bottom-right corner of the screen Toast Result Interface ■ Property Type Description action 'close' | 'retry' | 'action' | 'learn-more' | 'timeout' | string Action that triggered toast dismissal data any? Additional data from custom actions Custom Action Interface ■ Property Type Description action string Unique identifier for the action label string Display text for the action button data any? Additional data to pass with the action Best Practices ■ Design Guidelines ■ Choose appropriate variants - Use semantic colors that match the message type Keep messages concise - Toast notifications should be brief and actionable Use consistent positioning - Stick to one position for your application Consider duration carefully - Longer durations for important messages, shorter for confirmations Limit concurrent toasts - Avoid overwhelming users with too many notifications Accessibility ■ Provide meaningful titles - Use descriptive headers for screen readers Include action alternatives - Ensure all toast actions are keyboard accessible Test with screen readers - Verify proper announcement of toast content Maintain focus management - Ensure focus returns to appropriate elements Use sufficient contrast - Maintain readability in both light and dark themes Performance ■ Limit toast instances - Avoid creating excessive toast components Use appropriate timeouts - Set reasonable auto-dismiss durations Clean up resources - Ensure proper cleanup of event listeners Optimize animations - Use CSS transforms for smooth performance User Experience ■ Position strategically - Choose positions that don't block important content Provide clear actions - Make action buttons descriptive and intuitive Handle errors gracefully - Use error toasts with retry options when appropriate Consider mobile users - Ensure toasts are readable on small screens Respect user preferences - Allow users to control toast behavior Use Cases ■ Success

Confirmations - Operation completion, form submissions Error Notifications - Validation failures, API errors, network issues Warning Messages - Important reminders, confirmation requirements Information Updates - System status, feature announcements Action Prompts - User decisions, next steps, navigation hints Technical Notes ■ Service Architecture ■ The toast system uses a service-based architecture where: ToastService manages all toast operations and lifecycle ToastContainerComponent provides the positioning and container logic Individual toast components handle specific styling and behavior Dynamic component creation enables programmatic toast generation Positioning System ■ Toasts are positioned using CSS transforms and positioning: Fixed positioning ensures toasts appear above all content Responsive behavior adapts to different screen sizes Z-index management maintains proper layering Mobile optimization provides full-width toasts on small screens Animation System ■ Toast animations include: Entrance animations - Slide and fade in effects Exit animations - Slide and fade out effects Progress bars - Visual countdown for auto-dismiss Hover interactions - Timer pause/resume functionality Event Handling ■ The system handles multiple event types: Close events - User-initiated dismissal Action events - Button clicks and custom actions Timeout events - Auto-dismiss completion Hover events - Timer pause/resume functionality

```
<aava-button
  label="Show Success Toast"
  variant="success"
  (userClick)="showSuccessToast()"
></aava-button>
```

```
// Inject the toast service in your component
constructor(private toastService: ToastService) {}

// Show different types of toasts
showSuccessToast() {
  this.toastService.success({
    title: 'Successfully created!',
    message: 'Your changes have been saved successfully',
    duration: 2000,
    customWidth: '400px',
    design: 'modern',
    size: 'large',
  });
}
```

```

<aava-button
  label="Show Success Toast"
  variant="success"
  (userClick)="showSuccessToast()"
></aava-button>
<aava-button
  label="Show Error Toast"
  variant="danger"
  (userClick)="showErrorToast()"
></aava-button>
<aava-button
  label="Show Warning Toast"
  variant="warning"
  (userClick)="showWarningToast()"
></aava-button>
<aava-button
  label="Show Info Toast"
  variant="info"
  (userClick)="showInfoToast()"
></aava-button>
<aava-button
  label="Show Default Toast"
  variant="secondary"
  (userClick)="showDefaultToast()"
></aava-button>

```

```

// Inject the toast service in your component
constructor(private toastService: ToastService) {}

// Show different toast variants
showSuccessToast() {
  this.toastService.success({
    title: 'Successfully created!',
    message: 'Your changes have been saved successfully',
    duration: 2000,
    customWidth: '300px',
    design: 'modern',
    size: 'small',
  });
}

showErrorToast() {
  this.toastService.error({
    title: 'Error Occurred',
    message: 'Connection error. Unable to connect to the server at present',
    duration: 2000,
    customWidth: '300px',
    design: 'modern',
    size: 'small',
  });
}

showWarningToast() {

```

```

    this.toastService.warning({
      title: 'Warning Occurred',
      message: 'Please review your input carefully before proceeding.',
      duration: 2000,
      customWidth: '300px',
      design: 'modern',
      size: 'small',
    });
  }

  showInfoToast() {
    this.toastService.info({
      title: 'Action Required',
      message: 'Please review your input carefully before proceeding.',
      duration: 2000,
      customWidth: '300px',
      design: 'modern',
      size: 'small',
    });
  }

  showDefaultToast() {
    this.toastService.default({
      title: 'Default Toast Occurred',
      message: 'This is a default toast with neutral styling.',
      duration: 2000,
      customWidth: '300px',
      design: 'modern',
      size: 'small',
    });
  }

  showCustomToast() {
    this.toastService.custom({
      title: 'Custom Toast',
      message: 'This is a fully customizable toast with unique styling.',
      customBackground: 'var(--color-primary)',
      customTextColor: 'var(--color-text-primary)',
      customWidth: '300px',
      design: 'modern',
      size: 'small',
    });
  }
}

```

■ No code found